**WEB DEVELOPMENT**

**HTML**

Hyper Text Markup Language, Sir Tim Berners-Lee, 1993

We are going to learn HTML5.

**HEADING ELEMENT**

A heading element is used to write the heading.

It is represented by <h1>Content</h1>

The number inside the tag go up to 6 reducing in size i.e. 1 is the largest and 6 is the smallest heading.

NOTE: element is the whole <h1>Content</h1> thing and tag is just the <h1> (opening tag) and </h1> (closing tag)

**PARAGRAPH ELEMENT:**

A paragraph element is used to write paragraph. Two separate paragraphs have a line in between them.

It is represented by <p>this is a paragraph</p>

**HORIZONTAL RULE ELEMENT:**

A horizontal rule element is used to create a horizontal line between contents.

It is a Void element.

It can be represented by <hr /> or <hr>

It is recommended to use <hr /> over <hr>

**BREAK ELEMENT:**

A break element is used to break a paragraph from where it is used into a new line.

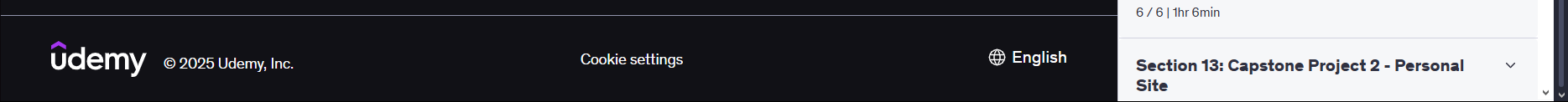
It is a void element.

It is represented by <br /> or <br>

It is recommended to use <br /> over <br>

**FOOTER ELEMENT:**

This element can be used to write a footer in out websites.



This is what a footer looks like.  
Example:

<footer>Name of company</footer>

**LIST ELEMENT:**

1. **UNORDERED LIST:**

It creates an unordered list (Like the list list not the one in python.)

To create an unordered list use <ul></ul> tags.

Between these another tag goes in i.e. <li></li>

Example:

<ul>

<li>Maggi</li>

<li>Paneer</li>

<li>Chicken</li>

</ul>

**OUTPUT:**

* Maggi
* Paneer
* Chicken

1. **ORDERED LIST:**

It creates an ordered list.

<ol>

<li>Maggi</li>

<li>Paneer</li>

<li>Chicken>/li>

</ol>

NOTE: creating a nested list results in indentations in the output.

NOTE: It is recommended to add indentation in your HTML code while creating list to increase the code readability.

**ANCHOR ELEMENT:**

This element helps us in creating hyperlinks (Texts that can be clicked to open another HTML file).

<a>This is another HTML file</a> #unactive hyperlink

We need to add an attribute for this element to work.

<a href = “link/location of file”>File</a> ad active Hyperlink

While executing it will look like this:

File

There are a lot more attributes that can be added to the opening tag for anchor element.

A global Attribute is an attribute that can be added to any element in HTML.

**IMAGE ELEMENT:**

This element adds an image into the website.

It takes **src** attribute for the link of the image.

<img src = “url”/>

It is a self-closing tag (Void element).

alt attribute is used to give the image a name which helps for screen readers to work on the website. (This attribute is also called alternative text description.)

**FILE PATHS:**

A file can be seen as the unique address if a file or a folder.

**1)ABSOLUTE FILE PATH:**

File path that is relative to the root folder.

Eg. C:/Projects/Bhabhi videos

**2)RELATIVE PATH:**

File path relative to the starting point where we are now.

../Maggi/Masala 🡪 .. can be used to consider a folder back as stariting point.

./Chappal/Ganda 🡪 . can be used to consider current folder as starting point.

**HTML BOILERPLATE:**

It is the structure of How an HTML file looks like.



1. <!DOCTYPE html> : this line tells the browser that the file is written in html code.
2. <html lang = “en”></html> : this element contains the actual HTML content. The lang attribute tells the browser that the page is going to be in English language.
3. <head></head> : This element contains some of the important details about our website which does not gets displayed to the user. CSS also goes here
4. <meta charset = “UTF-8”> : this meta tag tells the browser that UTF-8 character set has been used.
5. <title>Title</title> : This element contains the Title to be displayed in the tab of the website.
6. <body></body> : this element contains all the content to be displayed into the website.

NOTE: The whole boiler plate can be written by typing ! and smashing enter (Not that chupapi one).

**CSS**

Cascading Style sheet,1996,Håkon Wium Lie

CSS lessgo

There are three ways to add CSS file into the HTML website

1. Inline: <tag style = “css” />
2. Internal: <style>css</style>
3. External: <link href = “style.css”/>

**INLINE**

As the name suggests in this way, we write the style to be added into the tag. Below is an example, the highlighted part is the CSS code.

**Example:**

<html style = “background: blue”>

</html>

**INTERNAL**

In this way we use a separate tag between <head></head>. This tag is <style></style>. Below is an example, the highlighted part is the CSS code

**Example:**

<html>

<head>

<style>

html{

background: red;

}

</style>

</head>

</html>

Since we are applying this CSS on the html tag we gave its name in the style tag. This is called CSS selector (element selector).

This way of adding stylesheet is best for single pages.

**EXTERNAL**

In this way we create another file with .css extension and this file contains the code element. We also have to provide the link of this stylesheet inside the head tag in the project. Below is an example.

|  |  |
| --- | --- |
| Index.html | style.css |
| <html>  <head>  <link  rel = “stylesheet”  href = “./style.css”  />  </head>  </html> | html{  background: green;  } |

rel: relationship: this refers to what is the role of the thing we are linking this to.

href: This shows where the file is located at.(directory of the stylesheet)

This way of adding stylesheet is best for multipage websites.

**CSS SELECTORS**

1. **Element Selector:** This type of selector selects an element.

**Example:**

h1{

color: blue;

}

1. **Class Selector:** Selects the class of specified name. A class can be added to a tag as an attribute. <h1 class = “red-text”>

**Example:**

.red-heading{

color: red

}

1. **ID selector:** Selects a particular id specified. An id tag can be added as an attribute. <h1 id = “main”>

Note: You cannot have more than one element with the same id in an HTML document

**Example:**

#main{

color = green

}

1. **Attribute Selector:** Selects the tag with the specified attribute.

**Example:**

p[draggable]{

color = “red”

}

This selects all the paragraph elements with draggable attached to it.

1. **Universal Selector:** This selects everything.

**Example:**

\*{

color = “blue

}

**CSS Colors:**

We can specify the color type using two different ways, the first is using predefined names given to certain colors, and by specifying the hex code.

**Font Properties:**

1. Font size:

h1{

font-size: 20px;

}

1px = 1/96th inch

1pt = 1/72nd inch

1em = 100% of parent (defined as full width of letter m of parent. A parent can be considered as the tag it is enclosed in. say <h1> inside <body>)

1rem = 100% of root (defined as above but instead of parent it takes root i.e. html by default in browsers it is equivalent to 16px)

We can also use named font sizes.

1. Font weight:

h1{

font-weight: bold;

}

normal bold: keywords

lighter/bolder: Relative to Parent

number: 100-900

1. Font Family:

h1{

font-family: “Times New Roman”, sans-seriff

}

Here helvetica is a Type face.

Sans-seriff makes sure all the edges are at right angles.

We can find more fonts using google fonts and providing its link in the head under link tag.

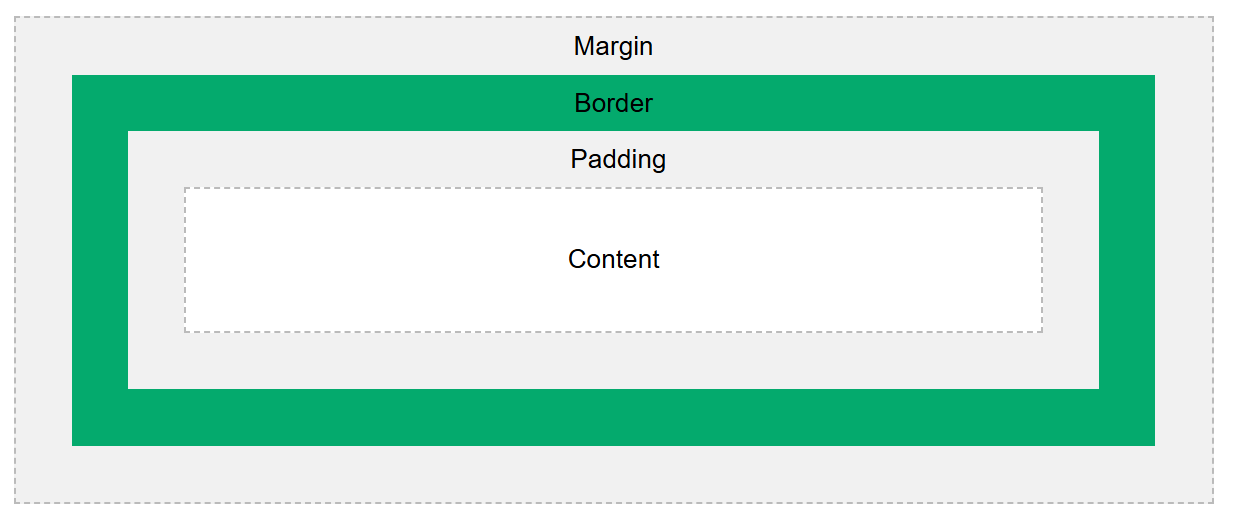
1. Text Allign:

h1{

text-align: centre;

}

**CSS Box Model:**

****

1. Border: This generates a border of specified width, style and color.

Example:

border: 10px solid black;

The border can be further modified by:

border-top: 0px (This changes the width of top border to 0px)

border-width: 0px 10px 20px 30px;

here the first is the top, right, bottom, left border.

1. Padding: this generates space between content and border.

Example:

padding: 20px;

1. Margin: This is the outside space between other content and the box.

Example:

margin: 10px;

Note: if only two values are specified into the width part

**Content Divison Element:**

This type of element is invisible while rendering. It is used to divide different piece of content and helps in managing elements by acting as an invisible box.

<div></div>

**The Cascade:**

In this we will learn about how the cascade works in CSS.

1. Position:

li{

color:red;

color: blue;

}

The color blue will be applied as it is after red. Li first becomes red then blue.

1. Specificity:

li {color: blue;}

.first-class {color: red;}

li[draggable] {color: purple;}

#first-id {color: orange;}

The importance goes from top to down. Here id is of the highest priority so the color is orange.

1. Type:

External 🡪 Internal 🡪 Inline

The increase in above order.

1. Importance:

color: red;

color: green !important;

we use !important to set the priority as high.

**Combining CSS Selectors:**

We can combine css selectors to target certain elements and reduce clutter in out Html code instead of writing everthing again and again.

Example:

.inner-box p{

color: white;

}

This targets all the paragraph elements that have parent class .inner-box

1. Group Rule: This applies the css code to all the selector group mentioned.

selector, selector{

color: blueviolet;

}

1. Child Rule: We can use this rule to select the child of a selector. This rule is only applied to selector one generation deep.

selector > selector {

color: firebrick;

}

1. Descendant Rule: Apply to descendent of left side. If the descendent selector is present in any generation of the parent selector the code os applied.

selector selector {

color: blue;

}

1. Chaining Rule: Apply where all selectors are True. We use this way to be very specific. We can add any numbers of selecotrs to chain.

selectorselector{

color: seagreen;

}

1. Combining combiners: We can combine the above combiners

selector selectorselector{

font-size: 2rem;

}

**CSS Positioning;**

As the name suggests we are going to learn to position the elements of out html in this section.

1. Static Positioning: This is applied by default.
2. Relative positioning: Position relative to default position.
3. Absolute positioning: Position relative to nearest positioned ancestor or top left corner of webpage if there is no ancestor.

NOTE: An attribute z-index can be used to specifiy the z-axis of of elements in html.

For Example: lets assume there are two colored square boxes red and green overlaping each other. If we specify the z-axis of red to be more than that of green then we will only be able to see the red box in the html.

While using Absolute positioning the html by default puts the box into another layer. If we put the z-index as -1 for the box it will go in the backside of the html. We cant take it back to the flow anymore though.

1. Fixed positioning: position relative to top left corner of the browser window. Even if we scroll up or down it will be there in the same position.

**CSS Shapes:**

1. Square: width = height
2. Rectangle: width != height
3. Circle: border-radius = 50%

**span():**

The <span> tag is an inline container used to mark up a part of a text, or a part of a document.

Example:

<p>My mother has <span style="color:blue">blue</span> eyes.</p>

Output:

My mother has blue eyes

**Display Property:**

1. Block: When display = block is set (is set by default for tags like <h1>,<p>) element takes up the whole of the width in html.
2. Inline: In this case instead of taking the whole of the width it takes up until only the part at which the text ends.
3. Inline-Block: same as inline but here we can set the height and the widht of the element.
4. None: It makes any element on the screen disappear.

**CSS Float and Clear:**

This helps in wrapping the text.

img{

float: left;

}

This will wrap the text on the left side of the image element just like in newspapers.

Note: For text to not be in wrapped (like footer) around the image. We have to apply another attribute clear to the footer.

footer{

clear: left;

}

**MAKING THE WEBSITE RESPONSIVE:**

**MEDIA QUERIES:**

This block of css code executes in places where the condition given in the bracket is true. This bracket condition is called breakpoint.

Example:

@media (max-width: 600px){

h1{

font-size: 15px;

}

}

1. max-width: This executes when the width is less than the specified resolution.
2. min-width: This executes when the width is more than the specified resolution.

Note: we can use and to combine two breakpoints.

Example:

@media (mix-width: 600px) and (max-width: 900px){

/\*Code\*/

}

This executes for devices with screen width less than 900px and more than 600px.

**FLEX BOX:**

Flexbox is a layout method for arranging items in rows or columns. Flexbox makes it easier to design a flexible responsive layout structure, without using float or positioning.

|  |  |
| --- | --- |
| **HTML** | **CSS** |
| <div class = “container”>  <div class = “one”><p>…</p></div>  <div class = “two”><p>…</p></div>  <div class = “three”><p>…</p></div>  </div> | .container{  display: flex;  gap: 10px;  } |

Floats are only good for wrapping texts around an image. For Creating the structure of the page we use flex box.

We can use inline-flex to combine flex and inline.

flex-direction: This changes the axis at which the flex box is directed.

flex-basis: This changes the axis length of the flex box.

Example: Say the flex-direction is set to row and flex basis is set to 100px. The row’s horizontal length will increase/decrease to 100px. If flex-direction is set to column and done the same, the height of the flex box increases.

Note: flex-basis is set on the child instead of on the parent.

order: This changes the order of the elements. The element with the highest value of order get in the last position. By default the order is 0 for all the elements.

flex-wrap: It sets whether flex items are forced onto new line or can wrap onto multiple lines. By default it is set to nowrap. It is set onto parent.

justify-content: This is used to allign the flex box.

align-items: This is used to allign items vertically. It needs another attribute height for that.

align-self: This is used to allign a singular element. This is specified onto the children.

align-content: This is used to allign elements when the size of the windowis changed.

flex-flow: This is used to combine flex-direction and flex-wrap as it occurs very often.

**Flex Sizing:**

The flex size priority is seen in the following order.

Content width < Width < flex-basis < min-width/max-width

max-width: This is the maximum width the flex will grow.

min-width: This is the maximum width the flex will shrink.

flex-grow: It allows the flex to grow past the flex basis size but not shrink less than it.

flex-shrink: It allows the flex to shrink less than the size of the flex basis and not grow past it.

Note: by default flex basis is set to auto. It refers to the content present in the flex box. The element with biggest word present gets the highest space and the one with the lowest gets the shortest. Instead of auto we write flex-basis: 0.

Note: To shorten flex-basis,grow,shrink we simply use flex: 0 0 0;

The first 0 sets the basis = 0

The second puts the grow = 0

The third puts the shrink = 0

We can define the ratios of flex boxes by defining flex: 1/2/3.

**GRID:**

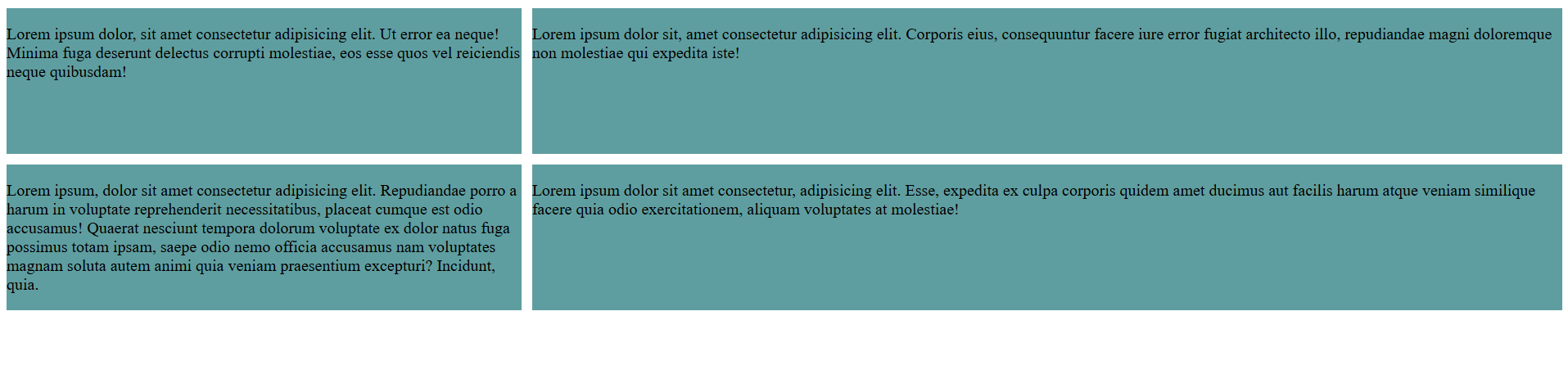
Flex box is best for horizontal allignment of items. Grid is best for alligning content in a 2D page. We use both in a page for making the process of creating a website more flexible. Grids line up with other elements.

We can create a grid by following.

|  |  |
| --- | --- |
| HTML | CSS |
| <div class = “container”>  <p>…</p>  <p>…</p>  <p>…</p>  <p>…</p>  </div> | .container{  display: grid;  grid-template-columns: 1fr 2fr;  grid-template-rows: 1fr 1fr;  gap: 10px;  } |

The above CSS can be overcome by using a shortened form

grid-template: 1fr 2fr/ 1fr 1fr;



**GRID SIZING:**

1. Fixed size: We can define the fixed size of grid as follows:-

grid-template-columns: 100px 200px;

Fixed sizing of grid makes irresponsive.

1. Auto size: We can put auto instead of a number to make it more responsive. The grid will extend to fit the content only.

grid-template-columns: auto;

1. Fractional size: We can use fractional values to define the ratio of sizes of the grids.

grid-template-rows: 1fr 2fr;

This makes the grid box responsive.

1. Minmax size: We can define the maximum and minimum width/height of the grid box by following.

grid-template-columns: 200px minmax(400px,800px)

1. Repear size: We can use repeat to define the size of multiple rows and column blocks without having to defining its height and width again and again.

grid-template-rows: repeat(2,200px);

Note: We can use grid-auto-rows/columns to define the size of rows and columns which in futute gets added into the grid gets the size.

**GRID PLACEMENT:**